CLAIM SET AS AMENDED

1. (Currently Amended) A method for processing data expressed as a class in JAVA ~~class~~ programming language to produce data expressed as an extensible markup language (XML) document, comprising:

loading ~~the~~ a named JAVA class;

determining if the loaded JAVA class implements a predefined interface, said predefined interface comprising annotations including a first parameter~~,~~ for associating a field of said JAVA class ~~field~~ with a corresponding XML element tag~~,~~ ; a second parameter~~,~~ for specifying a JAVA class to be instantiated when constructing said JAVA class field from said XML file~~,~~ ; a third parameter~~,~~ ; for identifying a JAVA method to invoke for retrieving said JAVA class field, and a fourth parameter~~,~~ for identifying a JAVA method to invoke for retrieving this method; and

in the case of said loaded JAVA class, implementing said predefined interface iteratively processing each field descriptor within the loaded JAVA class to retrieve a corresponding XML tag; and

transferring field values to new elements created using said corresponding XML tags.

2. (Currently Amended) A method for processing ~~an~~ data expressed as an extensible markup language (XML) document to produce a data expresses as a class in JAVA ~~class~~ programming language, comprising:

instantiating an object of ~~the~~ a desired JAVA class;

in the case of said instantiated object, implementing a predefined interface, iteratively processing each object included within said instantiated object according to the steps of:

retrieving field descriptors associated with ~~an~~ said object being processed;

creating an object of a specified JAVA type for each XML element corresponding to a field descriptor; and

storing the created object in the currently processed object.

3. (Currently Amended) A method for adapting ~~a~~ an object in JAVA ~~object~~ programming language to an application programming interface (API) for converting said JAVA object to data expressed in eXtensible Markup Language (XML), comprising:

annotating ~~said JAVA object to include, for each~~ said JAVA object to be converted to said XML to including the steps of~~,~~:

~~identification of~~ identifying a respective XML tag~~,~~ ;

~~identification of~~ identifying a JAVA class to be instantiated when constructing said JAVA object field from an XML file~~,~~ ;

~~identification of~~ identifying a JAVA method to invoke for retrieving said JAVA object; and

~~identification of~~ identifying a JAVA method to invoke for retrieving said retrieval method.

4. (Currently Amended) ~~An application programming interface (API) for~~ A method for converting at least date from JAVA programming language to data in extensible mark-up language (XML) using an application programming interface (API), the method comprising the steps of:

~~a field description retrieval method, for~~

retrieving a field description;

determining JAVA conversion parameters by examining an annotation associated with each JAVA element to be converted to XML, said annotation defining for each JAVA element at least a corresponding XML tag, a corresponding object class, a corresponding field retrieval method, and a corresponding method retrieval method.

5. (Currently Amended) The ~~API~~ method of claim 4, further comprising~~:~~ ~~a JAVAtoXML conversion method implemented according to~~ the steps of:

loading ~~the~~ a named JAVA class;

determining if ~~the~~ a loaded JAVA class implements a predefined interface, said predefined interface comprising annotations including:

a first parameter~~,~~ for associating a field of said JAVA class ~~field~~ with a corresponding XML element tag~~,~~ ;

a second parameter~~,~~ for specifying a JAVA class to be instantiated when constructing said JAVA class field from said XML file~~,~~ ;

a third parameter~~,~~ for identifying a JAVA method to invoke for retrieving said JAVA class field~~,~~ ; and

a fourth parameter, for identifying a JAVA method to invoke for retrieving this method; and

in the case of said loaded JAVA class, implementing said predefined interface iteratively processing each field descriptor within the loaded JAVA class to retrieve the corresponding XML tag; and

transferring field values to new elements created using said corresponding XML tags.

6. (Currently Amended) The method ~~API~~ of claim 4, further comprising:

~~an XMLtoJAVA conversion method, comprising:~~

instantiating an object of ~~the~~ a desired JAVA class;

in the case of said instantiated object implementing a predefined interface, iteratively processing each object included within said instantiated object according to the steps of:

retrieving field descriptors associated with an object being processed;

creating an object of a specified JAVA type for each XML element corresponding to a field descriptor; and

storing the created object in the currently processed object.

7. (Currently Amended) A ~~data structure~~ method for describing a class field in ~~a~~ JAVA programming language ~~class field~~ in a manner facilitating ~~XML~~ conversion to data expressed in extensible markup language (XML), comprising the steps of:

~~a first parameter, for~~ associating said JAVA class field with a corresponding XML element tag with a first parameter;

~~a second parameter, for~~ specifying a JAVA class to be instantiated when constructing said JAVA class field from said XML file with a second parameter;

~~a third parameter, for~~ identifying a JAVA method to invoke for retrieving said JAVA class field with a third parameter; and

~~a fourth parameter, for identifying a JAVA method to invoke for~~ retrieving ~~this method~~ the JAVA class field with a fourth parameter.

8. (Currently Amended) The ~~data structure~~ method of claim 7, further comprising:

~~a fifth parameter, for~~ specifying a type of JAVA object to instantiate for an XML element representing a collection with a fifth parameter.

9. (Currently Amended) The ~~data structure~~ method of claim 8, further comprising:

~~a sixth parameter, for~~ specifying a tag name to use for each element representing a collection with a sixth parameter.

10. (Currently Amended) The ~~data structure~~ method of claim 8, wherein said collection comprises a HashTable.

11. (New) A computer system for executing an application programming interface (API) function for converting data expressed as a class in JAVA programming language to data expressed in extensible mark-up language (XML), comprising:

processing means for executing the API function and memory means for storing the data to be converted,

wherein the API function includes a field description retrieval method for determining JAVA conversion parameters by examining an annotation associated with each JAVA element to be converted to XML, said annotation defining for each JAVA element at least a corresponding XML tag, a corresponding object class, a corresponding field retrieval method, and a corresponding method retrieval method.

12. (New) The computer system for executing an application programming interface (API) function of claim 11, further comprising to the steps of:

loading a named JAVA class into the processor;

determining if the loaded JAVA class implements a predefined interface, said predefined interface comprising annotations including a first parameter for associating a JAVA class field with a corresponding XML element tag, a second parameter for specifying

a JAVA class to be instantiated when constructing said JAVA class field from said XML file, a third parameter for identifying a JAVA method to invoke for retrieving said JAVA class field, and a fourth parameter for identifying a JAVA method to invoke for retrieving this method; and

in the case of said loaded JAVA class implementing said predefined interface, iteratively processing each field descriptor within the loaded JAVA class to retrieve corresponding XML tag; and

transferring field values to new elements created using said corresponding XML tags to the memory means of the computer system.

13. (New) The computer system for executing an application programming interface (API) function of claim 11, further comprising the steps of:

instantiating an object of the desired JAVA class;

in the case of said instantiated object implementing a predefined interface, iteratively processing each object included within said instantiated object according to the steps of:

retrieving field descriptors associated with an object being processed;

creating an object of specified JAVA type for each XML element corresponding to a field descriptor; and

storing the created object in the currently processed object.